3. Проектний розділ

В попередньому розділі було досліджено кілька шляхів побудови сховищ даних на основі платформи Hadoop. Результатом цього стало коротке порівняння двох наведених шляхів та структурні схеми двох наведених рішень.

В даному розділі буде спроектовано та детально описано список всіх компонентів системи та перелік технологій, обраних для реалізації. Потрібно брати до уваги факт, що технології в світі Hadoop розвиваються дуже стрімко і вже за короткий час весь перелік інструментів, обраних в даній роботі, може бути застарілим. Але навіть не зважаючи на це таке спроектовано рішення дозволить підбирати та оновлювати всі частини системи шляхом заміни компонентів на більш нові. Платформа спроектована так, що інтеграція та оновлення компонентів системи є доволі легким і не протребує багато роботи розробника.

Далі структурна схема, розроблена в попередньому розділі, буде розширена деталями про логіку роботи та руху даних. Така схема зможе слугувати вихідною точкою для безпосередньої розробки програмного рішення і виконувати роль карти на шляху побудови системи.

1. Проектування сховища даних

В попередньому розділі було розроблено структурну схему сховища даних, призначеного для виконання аналітичних запитів на побудови аналітичних звітів. Було наведено призначення кожної структурної частини системи та способои інтеграції частин між собою. Тепер мною буде запропоновано низькорівневий опис роботи системи та перелік технологій для побудови такого сховища даних.

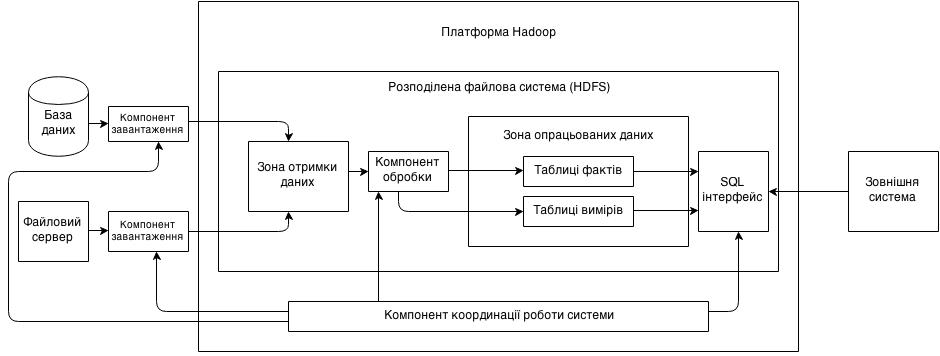


Рис. 3.1. Схема компонентів системи

В ході проектування мною буде обрано конкретні технології та інструменти для реалізації схеми наведеної на рис. 3.1. Як можна побачити, ця схема більш детально описує структуру системи, на відміну від схеми архітектури, наведеної в попередньому розділі. Основною різницею є наявність компонентів завантаження, обробки та координації роботи системи. Також в цій схемі детальніше наведено розмежування даних в розподіленій файловій системі. Таке розділення відображає логічні зони для даних, що існуватимуть в системі. В зоні отримання даних вся інформація зберігатиметься в такому вигляді, як вона надійшла з баз даних та файлових серверів. Згідно до структури побудови виміро-орієнтованих сховищ даних, всі дані будуть розбиті на таблиці фактів та таблиці вимірів, які і формуватимуть схему «зірка» або ж схему «сніжинка».

1. Вибір технологій для побудови системи

Оскільки система будуватиметься на основі платформи Hadoop, то одним з основих технологічних елементів, які потрібно обрати є дистрибутив Hadoop. Дистрибутивом платформи вважається пакет інтегрованих між собою та правильно налаштованих елементів екосистеми Hadoop. Сам по собі Hadoop це просто фреймворк для розподілених обчислень та розподілена файлова система. Всі елементи можна встановлювати самому, в ручному режимі, але це дуже ресурсоємкий процес, який потребує глибинних знань платформи. В даний момент є три найпоширеніших постачальника дистрибуцій: Hortonworks, Cloudera та MapR. З трьох описаних варіантів мною було вибрано постачальника Hortonworks. Ця компанія пропонує зовсім безплатно дистрибуцію HDP(Hortonworks Data Platform) 2.2. Цей набір включає в себе найновіші версії всіх основних компонентів екосистеми. Основними елементами, що входять до складу HDP 2.2 є:

* Hadoop & YARN(Yet another resource negotiator) 2.6.0
* Pig 0.14.0
* Hive & HCatalog 0.14.0
* HBase 0.98.4
* Spark 1.2.0
* Kafka 0.8.1
* Sqoop 1.4.5
* Ambari 1.7.0
* Oozie 4.1.0

Як видно зі списку – перелік технлогій доволі великий (а це тільки невелика частина), тому налаштувати їх вручну було б складно. Більшість наведених інстументів буде використовуватися при побудові системи. Далі буде наведено короткий опис призначення цих інструментів та спосіб, в який вони будуть використані при побудові сховища даних.

Основними компонентами, на яких власне й базуються всі обчислення в платформі є фреймворк Hadoop там менеджер ресурсів YARN. Це в першу чергу розподілена файлова система HDFS(Hadoop Distributed File System). Вона забезпечує надіє на ефективне зберігання великої кількості файлів на кластерах з великої кількості машин. Алгоритми розподілу даних в такій системі забезпечують велику швидкість доступу, читання та запису файлів. Файлова система також забезпечує високу найдійність за рахунок реплікакації даних. Менеджером ресурсів в платформі є компонент YARN. Він відповідає за розподіл виконання завдань по кластері з машин, та координує процес запуску, роботи, та збору результатів. Як HDFS так і YARN це набори процесів master та slave, які працюють на кожній з машин кластеру та спілкують з використанням мережних протоколів. Master-процеси виконують надсилання завданнь slave-процесам а також слідкують за їх виконанням. Всі обчислення, що відбувають в платформі Hadoop виконуються за посередництвом цих двох складових. Вони, на ряду з Map та Reduce завданнями є найбільш низькорівневими та фундаментальними блоками всіх Hadoop-систем.

Платформа Hadoop написана на мові Java та виконується як набір процесів у віртуальній машині Java. Це означає, що на цій мові можна написати будь яку програму, якщо в ній дотримана парадигма MapReduce, яка буде запускатися на кластері. Недоліком такого підходу є те, що як тільки логіка програми ставатиме порівняно тяжкою, буде надзвичайно важко відобразити її в термінах конструкцій мови Java. Для уникнення цих труднощів було розроблено кілька скриптових мов, які дозволяють набагато легше маніпулювати даними та виконувати їх перетворення. Однією з таких мов є скриптова мова Pig. В ній основними термінами є табличні структури даних та операції над ними. З допомогою цеї мови можна дуже легко будувати конвеєри обробки даних. Так з допомогої її конструкцій можна описати перелік трансформацій, які потрібно виконати над задекларованим набором даних, вказати джерело даних, місце та спосіб збереження та запустити на виконання. Транслятор цієї мови перетворить весь написаний код у набір Java MapReduce завдань, які будуть запущені на виконання в системі Hadoop. Використання такої мови значно пришвидшує і полегшує процес розробки. Особливо зручно використовувати такий інструмент, коли дані потребують достатньо складних перетворень, як наприклад зєднання по певному полю, перетин або фільтрація за певним критерієм. Процес розробки з використанням мови Pig є в кілька разів швидшим та потребує менше часу та зусиль на відлагодження.

Іншим схожим інструментом є Hive та HCatalog. Це дві взаємоповязані технології, що забезпечують зручний доступ до даних, що збережені в HDFS. HCatalog містить внутрішню базу даних з мета-інформацією про все, що знаходяться на HDFS. Ця інормація використовується при звертанні до даних та їх пошуку. Hive в свою чергу надає SQL-інтерфейс для структурованих даних в системі. Для забезпечення такого доступу, достатньо створити табицю в Hive, в які описати структуру даних та вказати місце розташування. Після цього ми зможемо легко звертатися до цих даних через SQL-інтерфейс. Як і у випадку Pig кожне звертання транслюється в набір Java MapReduce завдань та виконується на кластері. Виконання таких запитів значно поступається традиційним БД в швидкості, але це компенсовується обємами даних, на яких такі запити виконуються. На відміну від звичайних БД така система дозволяє виконувати складні SQL-запити на наборах даних, обємом в кілька десятків терабайт. Для оптимізації зберігання використовуються спеціальні формати з внутрішнім індексуванням та стисненням. На відміну від зберігання в звичайному текстовому форматі, таке зберігання дозволяє зменшити час доступу та обсяг необхідної для зберігання памяті в кілька десятків разів.

Іншим важливим компонентом, який, щоправда не буде використовуватися в цій реалізації системи є NOSQL база даних HBase. Вона забезпечує збереження та доступ майже в реальному часі до обємів даних в кілька десятків терабайт. Це база даних, в яій збереження даних організовано з групуванням по колонках. Характерною рисою її є те, що вона не потребує наперед заданої структури даних, що в ній зберігаються і підтримує її динамічну зміну. Через особливості внутрішньої організації, агрегуючі запити з використанням одної або кількох колонок в такій базі виконуються майже миттєво, незважаючи на обсяги інформації, що в ній збережена.